UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/728,441	12/01/2000	Wen-mei W. Hwu	042302 0269900	3226

27498 7590 01/30/2006

PILLSBURY WINTHROP SHAW PITTMAN LLP
P.O. BOX 10500
MCLEAN, VA 22102

EXAMINER	
LI, AIMEE J	
ART UNIT	**PAPER NUMBER**
2183	

DATE MAILED: 01/30/2006

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.	Applicant(s)	
	09/728,441	HWU ET AL.	
	Examiner	Art Unit	
	Aimee J. Li	2183	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *01 November 2005*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-50,52 and 53* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-50,52 and 53* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: _____.

DETAILED ACTION

1. Claims 1-50, 52, and new claim 53 have been considered. Claims 1, 3-8, 26, and 32 have been amended as per Applicant's request. Claim 51 has been cancelled as per Applicant's request.

Papers Submitted

2. It is hereby acknowledged that the following papers have been received and placed of record in the file: Amendment as received 01 November 2005.

Claim Rejections - 35 USC § 102

3. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (e) the invention was described in (1) an application for patent, published under section 122(b), by another filed in the United States before the invention by the applicant for patent or (2) a patent granted on an application for patent by another filed in the United States before the invention by the applicant for patent, except that an international application filed under the treaty defined in section 351(a) shall have the effects for purposes of this subsection of an application filed in the United States only if the international application designated the United States and was published under Article 21(2) of such treaty in the English language.

4. Claims 1-4 and 8-18 are rejected under 35 U.S.C. 102(e) as being taught by Mahalingaiah, U.S. Patent Number 5,989,865 (herein referred to as Mahalingaiah).

5. Referring to claim 1, Mahalingaiah has taught a processor comprising:

a. A functional unit adapted to execute an instruction issued to it from a dispatch stage (Mahalingaiah column 4, lines 32-43; column 11, lines 13-37; and Figure 1); and

b. A buffer in the dispatch stage coupled to the functional unit adapted to store a plurality of the instructions before issue to the functional unit and further adapted to store scheduling information associated with each of the instructions

(Mahalingaiah column 17, line 66 to column 18, line 16; and Figure 4). In regards to Mahalingaiah, the MROM Access stores all of the MROM instructions, which includes all loop instructions that includes the kernel instructions.

c. Wherein the plurality of the instructions comprise a kernel set of instructions from a loop body (Mahalingaiah column 2, lines 35-47; column 17, line 66 to column 18, line 16; and Figure 4). In regards to Mahalingaiah, all of the microcode loop instructions, including the kernel set of instructions in the loop, for a string MROM instruction are stored in the MROM Access.

d. Wherein the instructions are issued to the functional unit from the buffer in accordance with the stored scheduling information so as to cause the functional unit to execute a number of iterations of the body (Mahalingaiah column 17, line 66 to column 18, line 16; and Figure 4).

6. Referring to claim 2, Mahalingaiah has taught a decode stage register coupled between the dispatch stage and the functional unit, the functional unit coupled to the decode stage register for executing the instruction issued to the decode stage register from the dispatch stage (Mahalingaiah column 4, lines 32-44; column 9, lines 6-16; and Figure 1).

7. Referring to claim 3, Mahalingaiah has taught control logic coupled to the buffer adapted to cause a certain one of the stored plurality of instructions to be issued to the functional unit in accordance with a loop iteration stage (Mahalingaiah column 17, line 66 to column 18, line 16; column 19, lines 15-32; and Figure 4) and the stored scheduling information associated with the certain instructions (Mahalingaiah column 17, line 66 to column 18, line 16; and Figure 4).

8. Referring to claim 4, Mahalingaiah has taught wherein the control logic is further adapted to cause the certain one of the instructions to be issued in accordance with a cycle within the loop iteration stage (Mahalingaiah column 17, line 66 to column 18, line 16; column 19, lines 15-32; and Figure 4) and the stored scheduling information associated with the certain instructions (Mahalingaiah column 17, line 66 to column 18, line 16; and Figure 4).

9. Referring to claims 8, 10, 12, and 14, Mahalingaiah has taught

a. Control logic coupled to the buffer (Applicant's claim 8) (Mahalingaiah column 17, line 66 to column 18, line 16; column 19, lines 15-32; and Figure 4);

b. The control logic including:

i. A loop iteration register for storing a loop iteration parameter (Applicant's claims 8, 10, 12, and 14) (Mahalingaiah column 19, line 52 to column 20, line 5; column 22, line 49 to column 23, line 6; Figure 5; and Figure 7), and

ii. A loop cycles register for storing a loop cycles parameter (Applicant's claims 8, 10, 12, and 14) (Mahalingaiah column 19, line 52 to column 20, line 5; column 22, line 49 to column 23, line 6; Figure 5; and Figure 7);

c. Wherein the control logic is adapted to cause the plurality of instructions to be issued to the functional unit from the buffer in accordance with the loop iteration parameter, the loop cycles parameter and the stored scheduling information (Applicant's claim 8) (Mahalingaiah column 17, line 66 to column 18, line 16; column 19, lines 15-32; and Figure 4); and

d. The control logic including an iteration initiation register for storing a loop iteration initiation parameter (Applicant's claims 8, 10, 12 and 14) (Applicant's claims 8, 10, 12, and 14) (Mahalingaiah column 19, line 52 to column 20, line 5; column 22, line 49 to column 23, line 6; Figure 5; and Figure 7).

10. Referring to claims 9, 11, and 13, Mahalingaiah has taught wherein the control logic is operative so that the functional unit executes a number of loop iterations of the kernel set of loop instructions, a prologue set of loop instructions different than the kernel set of loop instructions, and an epilogue set of loop instructions different than the kernel set of loop instructions in accordance with the kernel set of loop instructions and received loop parameters (Applicant's claims 9, 11, and 13) (Mahalingaiah column 17, line 66 to column 18, line 16; and Figure 4). In regards to Mahalingaiah, the MROM Access stores all of the MROM instructions, which includes all loop instructions that includes the kernel instructions.

11. Referring to claims 15 and 16, Mahalingaiah has taught a fetch unit, wherein the control logic is further operative so that the fetch unit can be shut down during execution of a number of iterations of the loop body corresponding to the stored plurality of instructions (Mahalingaiah column 18, lines 58-60).

12. Referring to claims 17 and 18, Mahalingaiah has taught a fetch unit, wherein the control logic is further operative so that the fetch unit can be shut down during execution of the prologue, kernel, and epilogue sets of loop instructions (Mahalingaiah column 18, lines 58-60).

Claim Rejections - 35 USC § 103

13. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

14. Claims 5-7, 26-33, 35-41, and 43-49 rejected under 35 U.S.C. 103(a) as being unpatentable over Mahalingaiah, U.S. Patent Number 5,989,865 (herein referred to as Mahalingaiah) in view of Subramanian et al., U.S. Patent Number 5,867,711 (herein referred to as Subramanian).

15. Referring to claims 4-14, Mahalingaiah has not taught:

a. Wherein the scheduling information comprises a plurality of loop stage bit masks respectively associated with the plurality of instructions (Applicant's claim 5);

b. Wherein the scheduling information comprises a plurality of loop stage bit masks respectively associated with the plurality of instructions (Applicant's claims 6 and 7);

c. Wherein the control logic is further adapted to cause the certain one of the instructions to be issued in accordance with the respective one of the loop stage bit masks associated with the certain one of the instructions (Applicant's claims 6 and 7);

16. Subramanian has taught:

a. Wherein the buffer is further adapted to store a plurality of loop stage bit masks respectively associated with the plurality of instructions (Applicant's claim 5) (Subramanian column 2, lines 10-16; column 5, lines 29-45; column 10, lines 5-9; Figure 4; and Figure 5). In regards to Subramanian, there is a time-stamp, which

functions similarly to the stage bit masks, assigned to each instruction to denote when an instruction is to be executed.

b. Wherein the buffer is further adapted to store a plurality of loop stage bit masks respectively associated with the plurality of instructions (Applicant's claims 6 and 7) (Subramanian column 2, lines 10-16; column 5, lines 29-45; column 10, lines 5-9; Figure 4; and Figure 5).

c. Wherein the control logic is further adapted to cause the certain one of the instructions to be issued in accordance with the respective one of the loop stage bit masks associated with the certain one of the instructions (Applicant's claims 6 and 7) (Subramanian column 2, lines 10-16; column 5, lines 29-45; column 10, lines 5-9; Figure 4; and Figure 5). In regards to Subramanian, there is a time-stamp, which functions similarly to the stage bit masks, assigned to each instruction to denote when an instruction is to be executed.

17. In regards to Subramanian, the elements of the claims have been taught by Subramanian as necessary parts of modulo scheduling. A person of ordinary skill in the art at the time the invention was made would have recognized that modulo scheduling is a form of software pipelining specifically for loops, so that successive execution iterations are overlapped (Subramanian column 2, lines 8-10), thereby increasing processor efficiency and speed by executing multiple instructions at the same time. Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to incorporate the modulo scheduling of Subramanian in the device of Mahalingaiah to increase processor efficiency and speed.

18. Referring to claim 26, Mahalingaiah has taught a buffer in the dispatch stage of a processor, the buffer being associated with a functional unit that executes instructions issued to it from the dispatch stage and a first portion for storing a kernel set of loop instructions (Mahalingaiah column 4, lines 32-43; column 11, lines 13-37; Figure 1; column 17, line 66 to column 18, line 16; and Figure 4). Mahalingaiah has not taught a plurality of modulo schedule stage identifiers respectively associated with the kernel set of loop instructions, wherein the instructions are issued to the functional unit from the buffer in accordance with the stored modulo schedule stage identifiers so as to cause the functional unit to execute a number of iterations of a loop. Subramanian has taught a plurality of modulo schedule stage identifiers respectively associated with the kernel set of loop instructions, wherein the instructions are issued to the functional unit from the buffer in accordance with the stored modulo schedule stage identifiers so as to cause the functional unit to execute a number of iterations of a loop (Subramanian column 2, lines 10-16; column 5, lines 29-45; column 10, lines 5-9; Figure 4; and Figure 5). In regards to Subramanian, the elements of the claim have been taught by Subramanian as necessary parts of modulo scheduling. A person of ordinary skill in the art at the time the invention was made would have recognized that modulo scheduling is a form of software pipelining specifically for loops, so that successive execution iterations are overlapped (Subramanian column 2, lines 8-10), thereby increasing processor efficiency and speed by executing multiple instructions at the same time. Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to incorporate the modulo scheduling of Subramanian in the device of Mahalingaiah to increase processor efficiency and speed.

19. Referring to claims 27-31, Mahalingaiah has taught

a. Wherein the processor further includes control logic, the buffer being coupled to the control logic and the functional unit for causing the kernel set of loop instructions to be issued to the functional unit (Applicant's claim 27) (Mahalingaiah column 17, line 66 to column 18, line 16; column 19, lines 15-32; and Figure 4);

b. Wherein the loop instructions comprise undecoded instructions, and wherein the processor further includes a decode stage interposed between the functional unit and the buffer for decoding the instructions (Applicant's claim 30) (Mahalingaiah column 4, lines 32-44; column 9, lines 6-16; Figure 1; column 18, lines 50-57; and Figure 4); and

c. Wherein the loop instructions comprise decoded instructions in the form of functional unit control signals (Applicant's claim 31) (Mahalingaiah column 4, lines 32-44; column 9, lines 6-16; and Figure 1). In regards to Mahalingaiah, instructions are functional unit control signals since they control how the functional unit operates.

20. Mahalingaiah has not taught

a. The kernel set of loop instructions and modulo schedule stage identifiers (Applicant's claim 27);

b. Wherein the modulo schedule stage identifiers comprise bit fields (Applicant's claim 28);

c. The control logic determining whether to issue a certain one of the loop instructions to the functional unit in accordance with a bit position of a set bit in the bit field corresponding to the certain loop instruction (Applicant's claim 28); and

d. Wherein the control logic is operative to cause a number of loop iterations of the kernel set of loop instructions, a prologue set of loop instructions different than the kernel set of loop instructions, and an epilogue set of loop instructions different than the kernel set of loop instructions in accordance with the stored modulo schedule stage identifiers (Applicant's claim 29).

21. Subramanian has taught:

a. The kernel set of loop instructions and modulo schedule stage identifiers (Applicant's claim 27) (Subramanian column 2, lines 10-16; column 5, lines 29-45; column 10, lines 5-9; Figure 4; and Figure 5);

b. Wherein the modulo schedule stage identifiers comprise bit fields (Applicant's claim 28) (Subramanian column 3, lines 36-44);

c. The control logic determining whether to issue a certain one of the loop instructions to the functional unit in accordance with a bit position of a set bit in the bit field corresponding to the certain loop instruction (Applicant's claim 28) (Subramanian column 2, lines 10-16; column 4, lines 16-26; column 5, lines 29-45; column 10, lines 5-9; Figure 4; and Figure 5); and

d. Wherein the control logic is operative to cause a number of loop iterations of the kernel set of loop instructions, a prologue set of loop instructions different than

the kernel set of loop instructions, and an epilogue set of loop instructions different than the kernel set of loop instructions in accordance with the stored modulo schedule stage identifiers (Applicant's claim 29) (Subramanian column 2, lines 10-16; column 5, lines 29-45; column 6, lines 4-19; column 10, lines 5-9; Figure 4; and Figure 5).

22. In regards to Subramanian, the elements of the claims have been taught by Subramanian as necessary parts of modulo scheduling. A person of ordinary skill in the art at the time the invention was made would have recognized that modulo scheduling is a form of software pipelining specifically for loops, so that successive execution iterations are overlapped (Subramanian column 2, lines 8-10), thereby increasing processor efficiency and speed by executing multiple instructions at the same time. Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to incorporate the modulo scheduling of Subramanian in the device of Mahalingaiah to increase processor efficiency and speed.

23. Referring to claims 32 and 53, taking claim 32 as exemplary, Mahalingaiah has taught a processor for executing a number of iterations of a loop comprising:

a. A plurality of functional units (Mahalingaiah column 4, lines 32-43; column 11, lines 13-37; and Figure 1).

b. A dispatch stage coupled to the functional units for issuing instructions to the functional units (Mahalingaiah column 17, line 66 to column 18, line 16; column 19, lines 15-32; and Figure 4).

c. A plurality of buffers adapted to store loop instructions and scheduling information associated with the stored instructions (Mahalingaiah column 17, line 66 to column 18, line 16; and Figure 4). In regards to Mahalingaiah, a buffer to store a loop instruction has been taught and duplicating this part is not a patentable improvement. See *In re Harza* 274 F.2d 669, 124 USPQ 378 (CCPA 1960).

d. Control logic coupled to the plurality of buffers for causing the stored kernel set of instructions to be selectively issued to the functional units in accordance with the stored scheduling information (Mahalingaiah column 17, line 66 to column 18, line 16; column 19, lines 15-32; and Figure 4).

24. Mahalingaiah has not taught

a. The loop including a prologue set of loop instructions, a kernel set of loop instructions and an epilogue set of loop instructions; and

b. The control logic being operative so that the functional units execute the number of iterations of the kernel set of loop instructions, the prologue set of loop instructions and the epilogue set of loop instructions based on the stored kernel set of loop instructions and associated scheduling information.

25. Subramanian has taught

a. The loop including a prologue set of loop instructions, a kernel set of loop instructions and an epilogue set of loop instructions (Subramanian column 6, lines 4-19 and Figure 5); and

b. The control logic being operative so that the functional units execute the number of iterations of the kernel set of loop instructions, the prologue set of loop instructions and the epilogue set of loop instructions based on the stored kernel set of loop instructions (Subramanian column 2, lines 10-16; column 5, lines 29-45; column 10, lines 5-9; Figure 4; and Figure 5).

26. In regards to Subramanian, the elements of the claim have been taught by Subramanian as necessary parts of modulo scheduling. A person of ordinary skill in the art at the time the invention was made would have recognized that modulo scheduling is a form of software pipelining specifically for loops, so that successive execution iterations are overlapped (Subramanian column 2, lines 8-10), thereby increasing processor efficiency and speed by executing multiple instructions at the same time. Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to incorporate the modulo scheduling of Subramanian in the device of Mahalingaiah to increase processor efficiency and speed.

27. Claim 53 is functionally similar to claim 32 and is rejected for the same reasons set forth above.

28. Referring to claim 33, Mahalingaiah has taught a fetch unit, wherein the control logic is further operative so that the fetch unit can be shut down during execution of the number of iterations of the loop (Mahalingaiah column 18, lines 58-60).

29. Referring to claims 35 and 43, Mahalingaiah has taught a method for executing a number of iterations of a loop in a processor, the method comprising:

a. Storing the kernel set of loop instructions at a dispatch stage of the processor (Mahalingaiah column 17, line 66 to column 18, line 16; and Figure 4); and

b. Storing loop parameters in control logic associated with the stored loop instructions (Mahalingaiah column 17, line 66 to column 18, line 16; and Figure 4).

30. Mahalingaiah has not taught:

a. The loop including a prologue set of loop instructions, a kernel set of loop instructions and an epilogue set of loop instructions; and

b. Causing the stored kernel set of instructions to be selectively issued to functional units of the processor in accordance with the stored loop parameters so that the functional units of the processor execute the number of iterations of the kernel set of loop instructions, the prologue set of loop instructions and the epilogue set of loop instructions based on the stored loop instructions.

31. Subramanian has taught:

a. The loop including a prologue set of loop instructions, a kernel set of loop instructions and an epilogue set of loop instructions (Subramanian column 6, lines 4-19 and Figure 5); and

b. Causing the stored kernel set of instructions to be selectively issued to functional units of the processor in accordance with the stored loop parameters so that the functional units of the processor execute the number of iterations of the kernel set of loop instructions, the prologue set of loop instructions and the epilogue set of

loop instructions based on the stored loop instructions (Subramanian column 2, lines 10-16; column 5, lines 29-45; column 10, lines 5-9; Figure 4; and Figure 5).

32. In regards to Subramanian, the elements of the claim have been taught by Subramanian as necessary parts of modulo scheduling. A person of ordinary skill in the art at the time the invention was made would have recognized that modulo scheduling is a form of software pipelining specifically for loops, so that successive execution iterations are overlapped (Subramanian column 2, lines 8-10), thereby increasing processor efficiency and speed by executing multiple instructions at the same time. Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to incorporate the modulo scheduling of Subramanian in the device of Mahalingaiah to increase processor efficiency and speed.

33. Referring to claims 36-40 and 44-48, Mahalingaiah has taught

a. A loop iteration register for storing a loop iteration parameter (Applicant's claims 36 and 44) (Mahalingaiah column 19, line 52 to column 20, line 5; column 22, line 49 to column 23, line 6; Figure 5; and Figure 7), and

b. A loop cycles register for storing a loop cycles parameter (Applicant's claims 36 and 44) (Mahalingaiah column 19, line 52 to column 20, line 5; column 22, line 49 to column 23, line 6; Figure 5; and Figure 7);

34. Mahalingaiah has not taught:

a. The control logic including an iteration initiation register for storing a loop iteration initiation parameter (Applicant's claims 36 and 44);

b. Adjusting the value in the loop cycles register from the stored loop cycles parameter in accordance with a processor cycle (Applicant's claims 37 and 45);

c. Resetting the value in the loop cycles register in accordance with the adjusted value and the stored iteration initiation parameter (Applicant's claims 37 and 45);

d. Adjusting the value in the loop iteration register in accordance with the resetting step (Applicant's claims 38 and 46);

e. Completing the issue of loop instructions in accordance with the adjusted value in the loop iteration register (Applicant's claims 38 and 46);

f. Storing a set of modulo schedule stage identifiers respectively associated with the kernel set of loop instructions, the step of causing the stored kernel set of loop instructions to be selectively issued including the step of comparing the stored loop parameters with the modulo schedule stage identifiers (Applicant's claims 39 and 47); and

g. Wherein the step of comparing the stored loop parameters with the modulo schedule stage identifiers includes the step of indexing to a certain one of the modulo schedule stage identifiers in accordance with a current cycle in the modulo schedule stage indicated by the stored loop parameters, the step of causing the stored kernel set of loop instructions to be selectively issued further including issuing the associated loop instruction to the functional unit if the certain modulo schedule stage identifier indicates that the functional unit is active (Applicant's claims 40 and 48).

35. Subramanian has taught:

a. The control logic including an iteration initiation register for storing a loop iteration initiation parameter (Applicant's claims 36 and 44) (Subramanian column 5, lines 49-56 and Figure 5);

b. Adjusting the value in the loop cycles register from the stored loop cycles parameter in accordance with a processor cycle (Applicant's claims 37 and 45) (Subramanian column 5, line 29 to column 7, line 7; Figure 4; and Figure 5);

c. Resetting the value in the loop cycles register in accordance with the adjusted value and the stored iteration initiation parameter (Applicant's claims 37 and 45) (Subramanian column 5, line 29 to column 7, line 7; Figure 4; and Figure 5);

d. Adjusting the value in the loop iteration register in accordance with the resetting step (Applicant's claims 38 and 46) (Subramanian column 5, line 29 to column 7, line 7; Figure 4; and Figure 5);

e. Completing the issue of loop instructions in accordance with the adjusted value in the loop iteration register (Applicant's claims 38 and 46) (Subramanian column 5, line 29 to column 7, line 7; Figure 4; and Figure 5);

f. Storing a set of modulo schedule stage identifiers respectively associated with the kernel set of loop instructions, the step of causing the stored kernel set of loop instructions to be selectively issued including the step of comparing the stored loop parameters with the modulo schedule stage identifiers (Applicant's claims 39 and 47) (Subramanian column 2, lines 10-16; column 5, lines 29-45; column 10, lines 5-9; Figure 4; and Figure 5); and

g. Wherein the step of comparing the stored loop parameters with the modulo schedule stage identifiers includes the step of indexing to a certain one of the modulo schedule stage identifiers in accordance with a current cycle in the modulo schedule stage indicated by the stored loop parameters, the step of causing the stored kernel set of loop instructions to be selectively issued further including issuing the associated loop instruction to the functional unit if the certain modulo schedule stage identifier indicates that the functional unit is active (Applicant's claims 40 and 48) (Subramanian column 5, line 29 to column 7, line 7; Figure 4; and Figure 5).

36. In regards to Subramanian, the elements of the claims have been taught by Subramanian as necessary parts of modulo scheduling. A person of ordinary skill in the art at the time the invention was made would have recognized that modulo scheduling is a form of software pipelining specifically for loops, so that successive execution iterations are overlapped (Subramanian column 2, lines 8-10), thereby increasing processor efficiency and speed by executing multiple instructions at the same time. Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to incorporate the modulo scheduling of Subramanian in the device of Mahalingaiah to increase processor efficiency and speed.

37. Referring to claims 41 and 49, Mahalingaiah has taught shutting down the fetch unit during execution of the number of iterations of the loop (Mahalingaiah column 18, lines 58-60).

38. Claims 19-21 are rejected under 35 U.S.C. 103(a) as being unpatentable over Mahalingaiah as applied to claims 9, 11, and 13 above, in view of Valluri and Govindarajan's

"Modulo-Variable Expansion Sensitive Scheduling" published in High Performance Computing, 1998 (herein referred to as Valluri). Mahalingaiah in view of Subramanian has not taught wherein the kernel set of loop instructions comprise MVE code. Valluri has taught wherein the kernel set of loop instructions comprise MVE code (Valluri section 1. Introduction, paragraphs 1-2). A person of ordinary skill in the art at the time the invention was made would have recognized that MVE is needed to handle overlapping of a single variable with a subsequent definition of itself by ensuring that different registers are used each time (Valluri section 1. Introduction, paragraphs 1-2). Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to incorporate MVE of Valluri in the device of Mahalingaiah in view of Subramanian to handle same variable overlap.

39. Claims 22-25, 34, 42, and 50 are rejected under 35 U.S.C. 103(a) as being unpatentable over Mahalingaiah in view of Subramanian as applied to claims 3, 8, 11, and 13 above, and further in view of Mason et al., U.S. Patent Number 6,418,489 (herein referred to as Mason). Mahalingaiah has not taught wherein the control logic is further operative to allow interrupts to be handled at the end of a current loop iteration. Mason has taught wherein the control logic is further operative to allow interrupts to be handled at the end of a current loop iteration and to complete the number of loop iterations after the interrupt is handled (Mason column 6, lines 32-33). In regards to Mason, returning to complete the loop iterations is inherent to interrupts, since they are only temporary. Please see InstantWeb's Online Computing Dictionary. A person of ordinary skill in the art at the time the invention was made would have recognized that waiting until the end of a loop iteration to allow interrupts would ensure data is not lost and/or corrupted and continuing afterwards ensures the process completes and normal process has resumed.

Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to incorporate the interrupts of Mason in Mahalingaiah.

40. Claim 53 is rejected under 35 U.S.C. 103(a) as being unpatentable over Mahalingaiah, U.S. Patent Number 5,989,865 (herein referred to as Mahalingaiah), as applied to claim 1 above, in view of Fleck et al., U.S. Patent Number 6,076,159 (herein referred to as Fleck). Mahalingaiah has taught a plurality of other functional units (Mahalingaiah column 4, lines 32-43; column 11, lines 13-37; and Figure 1) and wherein the buffer is coupled to the functional unit and adapted to that it store instruction for execution by the functional unit (Mahalingaiah column 4, lines 32-43; column 11, lines 13-37; and Figure 1). Mahalingaiah has not explicitly taught instructions for execution by the functional unit but not for any of the other functional units. However, Mahalingaiah has taught multiple functional units in general but not their specific operation. Fleck has taught instructions for execution by the functional unit but not for any of the other functional units (column 1, lines 39-62; column 2, lines 26-39; column 2, line 56 to column 3, line 17; column 3, lines 30-40; Figure 1; and Figure 3). In regards to Fleck, there are multiple functional units each with specific purposes, including one unit that specifically executes loops only, so that each unit only executes a particular type of instruction(s). A person of ordinary skill in the art at the time the invention was made, and as taught by Fleck, would have recognized that having specific functions for each functional unit minimizes dependency analysis and control needed for the simultaneously executing pipelines, thereby increasing execution speed without the necessity of including the hardware of dependency analysis and control (Fleck column 1, lines 39-62 and column 2, lines 31-40). Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to incorporate

the specific functional units of Fleck in the device of Mahalingaiah to increase execution speed while minimizing hardware additions.

Response to Arguments

41. Applicant's arguments filed 01 November 2005 have been fully considered but they are not persuasive.

42. Applicant argues in essence on pages 13-14

> ...Accordingly, Mahalingaiah does not disclose or suggest that MROM Access stores scheduling information associated with each of the instructions... Moreover, Mahalingaiah does not disclose or suggest that instructions from MROM Access are issued to the functional unit from the buffer in accordance with the stored scheduling information so as to cause the functional unit to execute a number of iterations of the loop body...

43. This has not been found persuasive. The claim language is specifically "a buffer...to store scheduling information associated with each of the instructions...wherein the instructions are issued to the functional unit from the buffer in accordance with the stored scheduling information so as to cause the functional unit to execute a number of iterations of the body." According to the claim language all that is needed is for scheduling information associated with loop instruction(s) to be stored in a buffer. Mahalingaiah shows in Figure 4 that the MROM Unit contains a sequence control portion and Figure 6 shows a counter and comparator to control the loop iterations. Additionally, Mahalingaiah shows in Figure 7 a flow chart teaching how Mahalingaiah controls loops, which includes deciding whether the loop has executed the proper number of times. Also, merely shifting the scheduling information from outside the buffer to

into the buffer is not a patentable change. See *In re Japikse*, 181 F.2d 1019, 86 USPQ 70 (CCPA 1950).

44. Applicants argues in essence on pages 14-15 "...It does not determine addresses of specific lines associated with a set of MROM microinstructions in MROM Access based on loop iteration stage, much less addresses of certain one of the microinstructions." This has not been found persuasive. The exact claim language is

> control logic coupled to the buffer adapted to cause a certain one of the stored plurality of instructions to be issued to the functional unit in accordance with a loop iteration stage and the stored scheduling information associated with the certain instruction.

45. The claim language, when taken in the broadest reasonable interpretation, merely means that an instruction is issued from the buffer depending on the loop iteration stage and scheduling information. The loop iteration stage, in the broadest reasonable interpretation, merely means whether the loop is executing or not. The scheduling information, in the broadest reasonable interpretation, can mean which loop iteration the loop is on. Or vice versa. Mahalingaiah executes the MROM instructions dependent upon whether the loop is still executing or not and the loop iteration, as is shown in Figure 7. Applicants' arguments seem to insinuate that there are definitions to these phrases that are not explicitly defined in the specification or claim language. In response to applicant's argument that the references fail to show certain features of applicant's invention, it is noted that the features upon which applicant relies (i.e., loop iteration stage and scheduling information) are not recited in the rejected claim(s). Although the claims

are interpreted in light of the specification, limitations from the specification are not read into the claims. See *In re Van Geuns*, 988 F.2d 1181, 26 USPQ2d 1057 (Fed. Cir. 1993).

46. Applicant argues in essence on page 15 "...The invention...requires and measure of the number of cycles...in the loop body..." This has not been found persuasive. The claim language in question, in the broadest reasonable interpretation, can mean a cycle when the loop is still executing, e.g. where the number of loop iterations is still being tracked. Applicants' arguments seem to be arguing a narrower definition of the phrase "a cycle within the loop iteration stage" than is explicitly defined in the specification and in the claim. In response to applicant's argument that the references fail to show certain features of applicant's invention, it is noted that the features upon which applicant relies (i.e., a cycle within the loop iteration stage) are not recited in the rejected claim(s). Although the claims are interpreted in light of the specification, limitations from the specification are not read into the claims. See *In re Van Geuns*, 988 F.2d 1181, 26 USPQ2d 1057 (Fed. Cir. 1993).

47. Applicant argues in essence on page 15-16 "...requires a measure of the number of cycles...in the loop body, as well as the loop iteration initiation..." This has not been found persuasive. As shown by Mahalingaiah in Figure 7, Mahalingaiah needs three pieces of information to determine if the loop can be exited or cancelled: the string count, number of iterations of the microcode loop, and whether the two are greater than or equal to each other. These are the loop iteration initiation parameter, loop iteration parameter, and the loop cycles parameter. The initiation parameter is the string count, since a string that requires a loop starts a loop. The loop iteration parameter is the number of iterations the microcode loop has executed, since that is the loop iteration currently being executed. The loop cycles parameter is the

comparison result, since it represents whether the loop cycle is complete or not. Applicants' arguments seem to be suggesting a meaning to these terms that is not reflected in the claim language nor is there an explicit definition within the specification. In response to applicant's argument that the references fail to show certain features of applicant's invention, it is noted that the features upon which applicant relies (i.e., the loop iteration initiation parameter, loop iteration parameter, and the loop cycles parameter) are not recited in the rejected claim(s). Although the claims are interpreted in light of the specification, limitations from the specification are not read into the claims. See *In re Van Geuns*, 988 F.2d 1181, 26 USPQ2d 1057 (Fed. Cir. 1993).

48. Applicant argues in essence on pages 16-17

> ...Mahalingaiah especially does not disclose control logic that is operative to such that the functional unit executes the number of iterations of the kernel set of loop instructions, the **prologue** set of loop instructions **and** the **epilogue** set of loop instructions **based on the stored kernel loop instructions** and the received loop parameters...

49. This has not been found persuasive. Mahalingaiah has taught that the MROM instructions are all stored in the same location. This includes the kernel, prologue, and epilogue instructions. As Mahalingaiah shows in Figures 4, 6, and 7, the determination of whether the loop is executed again, including whether the kernel, prologue, and epilogue, is based upon the number of iterations the microcode loop has executed, which includes whether the number of times the kernel instructions have been executed, and this is reflected in the loop parameters. This means that the execution of the loop in the functional unit is based upon the loop

parameters, which reflects the information about the kernel instruction iterations, i.e. based on the kernel set of loop instructions.

50. Applicant argues in essence on pages 17 and 23 "...control logic is operative so that the **fetch unit can be shut down**..." This has not been found persuasive. Stalling instructions is functionally similar to shutting down the fetch unit for the breadth of the claim. Shutting down a fetch unit merely means that instructions are no longer fetched. When a fetch unit is stalled, the instructions are no longer fetched.

51. Applicant argues in essence on pages 18-22

> Moreover, as set forth more fully above, Subramanian teaches **compiler software** methods for creating a modulo schedule of a loop iteration, and thus **teaches away** from, providing any modulo scheduling control logic **in a processor**...
> ...motivation for any change of Mahalingaiah's **structure** must come from Subramanian's **own teachings**...

52. This has not been found persuasive. There is nothing in Subramanian explicitly stating that Subramanian's method cannot be used in a system such as Mahalingaiah's. Also, whether Subramanian teaches software or hardware is irrelevant. Whatever can be executed in software can be implemented in hardware. Please see Tanenbaum's Structured Computer Organization ©1984. The current claim language does not set apart the hardware implementation of the arguments from the software implementation taught in Subramanian. In response to applicant's argument that the compiler software teaches away from the invention, the test for obviousness is not whether the features of a secondary reference may be bodily incorporated into the structure of the primary reference; nor is it that the claimed invention must be expressly suggested in any

one or all of the references. Rather, the test is what the combined teachings of the references would have suggested to those of ordinary skill in the art. See *In re Keller*, 642 F.2d 413, 208 USPQ 871 (CCPA 1981). Also, the motivation to combine does not need to come from Subramanian's own teachings. It merely needs to be apparent to a person of ordinary skill in the art and the extrinsic information cited was to show that the reason to incorporate something like Subramanian was know to a person of ordinary skill in the art. In response to applicant's argument that there is no suggestion to combine the references, the examiner recognizes that obviousness can only be established by combining or modifying the teachings of the prior art to produce the claimed invention where there is some teaching, suggestion, or motivation to do so found either in the references themselves or in the knowledge generally available to one of ordinary skill in the art. See *In re Fine*, 837 F.2d 1071, 5 USPQ2d 1596 (Fed. Cir. 1988)and *In re Jones*, 958 F.2d 347, 21 USPQ2d 1941 (Fed. Cir. 1992).

53. Applicant argues in page 24, "...In modulo scheduled code according to the present invention, there is not natural and clean end of a kernel iteration; all the iterations are overlapped..." This has not been found persuasive. Mahalingaiah shows in Figure 7 that, when the comparison shows that the iteration number is greater than or equal to the string count, a cancel signal is asserted or the loop is somehow exited. This is normally done via an interrupt, however, Mahaligaiah has not explicitly taught the interrupt. Mason was relied upon to show that interrupts are common ways of executing this type of function. n response to applicant's argument that Mason's interrupt are not proper for the claimed function, the test for obviousness is not whether the features of a secondary reference may be bodily incorporated into the structure of the primary reference; nor is it that the claimed invention must be expressly suggested in any

one or all of the references. Rather, the test is what the combined teachings of the references would have suggested to those of ordinary skill in the art. See *In re Keller*, 642 F.2d 413, 208 USPQ 871 (CCPA 1981). Also, Applicants' arguments are focusing on an interrupt at the end of the kernel, but there is no language in the claim regarding the kernel set of the instructions with regard to the interrupt. In response to applicant's argument that the references fail to show certain features of applicant's invention, it is noted that the features upon which applicant relies (i.e., interrupt at the end of a kernel set of instructions) are not recited in the rejected claim(s). Although the claims are interpreted in light of the specification, limitations from the specification are not read into the claims. See *In re Van Geuns*, 988 F.2d 1181, 26 USPQ2d 1057 (Fed. Cir. 1993).

Conclusion

54. The prior art made of record and not relied upon is considered pertinent to applicant's disclosure as follows. Applicant is reminded that in amending in response to a rejection of claims, the patentable novelty must be clearly shown in view of the state of the art disclosed by the references cited and the objections made. Applicant must also show how the amendments avoid such references and objections. See 37 CFR § 1.111(c).

a. Park, U.S. Patent Number 6,988,190, has taught a cache containing loop information.

55. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Aimee J. Li whose telephone number is (571) 272-4169. The examiner can normally be reached on M-T 7:30am-5:00pm.

56. If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Eddie Chan can be reached on (571) 272-4162. The fax phone number for the organization where this application or proceeding is assigned is 703-872-9306.

57. Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

AJL
Aimee J. Li
21 January 2006

EDDIE CHAN
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100